Domain-Abstraction Heuristics for Simple Numeric Planning

Markus Fritzsche[1], **Mikhail Gruntov**[2], **Alexander Shleyfman**[3], **Daniel Gnad**[4,1]

[1]Linköping University, Sweden
[2]Technion, Israel
[3]Bar-Ilan University, Israel
[4]Heidelberg University, Germany
markus.fritzsche@liu.se, gruntovm@campus.technion.ac.il, alexash@biu.ac.il, daniel.gnad@uni-heidelberg.de

Abstract

Abstraction heuristics are among the most effective approaches in optimal classical planning. For numeric planning, however, existing abstraction heuristics suffer from the infiniteness of the abstract state spaces, which is an immediate consequence of numeric state variables. This has recently been analyzed for numeric Pattern Database (PDB) heuristics, which fall short of their classical-planning counterpart due to fundamental limitations in the handling of unbounded variable domains. In this work, we argue that domain abstractions offer a framework that lends itself much better to simple numeric planning, with abstract state spaces that are computed incrementally using counterexample-guided abstraction refinement (CEGAR), avoiding the exhaustive exploration of PDBs. We extend the established framework from classical planning such that the typically infinite concrete state space is fully represented in the abstraction, and adapt the CEGAR mechanism to support refining numeric abstractions. To obtain a strong search guidance, we combine multiple domain abstractions admissibly using the canonical heuristic. Our empirical evaluation exemplifies the potential of domain abstractions for numeric planning.

Code — https://github.com/dgnad/numeric-fast-downward
Datasets — https://doi.org/10.5281/zenodo.20424990

Introduction

Optimal planning in deterministic environments relies critically on informative admissible heuristics to guide an A^* state-space search (Hart, Nilsson, and Raphael 1968; Pearl 1984). In classical planning, abstraction heuristics – most prominently pattern databases (PDBs) (Culberson and Schaeffer 1998; Edelkamp 2001) and generalizations such as Cartesian and Merge&Shrink abstractions (Helmert et al. 2014; Seipp and Helmert 2018) – have established themselves as state-of-the-art admissible heuristics. A key factor in their success is the construction of abstractions that are both informative and efficiently computable.

Counterexample-guided abstraction refinement (CEGAR), originally proposed for model checking (Clarke et al. 2000), has proven particularly effective for generating high-quality abstractions in classical planning: it has been applied to PDB generation (Rovner, Sievers, and Helmert 2019) and to Cartesian abstraction refinement (Seipp and Helmert 2018). Recently, Kreft et al. (2023) demonstrated that domain abstractions – positioned between PDBs and Cartesian abstractions in terms of their expressive power – can be efficiently generated by CEGAR and outperform both PDBs and Cartesian abstractions in classical planning. Büchner et al. (2024) subsequently extended this CEGAR-based framework to the more general factored-task setting.

Despite these advances, abstraction heuristics have been almost exclusively limited to finite-domain planning. Numeric planning poses significant challenges for abstractions. Even in the simplest case – integer-valued state variables, linear numeric conditions, and simple numeric effects that add or subtract constants – planning tasks induce infinite transition systems, and plan existence is undecidable (Helmert 2002; Gnad et al. 2023). Existing abstraction heuristics for numeric planning – numeric PDBs and interval-based effect abstractions (Li et al. 2018) – suffer from several limitations: (1) inefficient implementation due to floating-point values and unbounded variable domains; (2) no heuristic on abstract states not explored during PDB construction; and (3) the lack of a scalable and targeted PDB exploration to identify important regions in the abstract state space (Gnad et al. 2025; Fritzsche et al. 2026).

In this work, we introduce a CEGAR-based abstraction framework for *simple numeric planning* (SNP) (Scala et al. 2020). We adopt the CEGAR pipeline that has proven successful in classical planning (Seipp 2017; Seipp, Keller, and Helmert 2020) and extend it to numeric tasks with unbounded integer-valued variables and infinite transition systems. Our contributions are: (1) we generalize domain abstractions (Kreft et al. 2023) to SNP, introducing abstraction components capable of partitioning infinite integer domains without relying on floating-point storage or enumerating finite subsets; (2) we develop a numeric CEGAR refinement procedure that efficiently derives counterexamples across infinite successor spaces. The refinement process selectively splits numeric domains into intervals to eliminate infeasible abstract counterexamples while guaranteeing termination under natural constraints; (3) we show how to admissibly combine these abstractions using the canonical heuristic. We compare our approach empirically to the numeric variants of PDBs and LM-cut (Kuroiwa et al. 2022) on established simple numeric planning benchmarks.

Preliminaries

We adopt the restricted numeric planning (RT) (Hoffmann 2003) formalism, which is as expressive as simple numeric planning (SNP). Here, a task is given as a tuple $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, G \rangle$, where $\mathcal{V} = \mathcal{V}_n \cup \mathcal{V}_p$ denotes the set of *numeric* and *finite-domain variables*, $\mathcal{A}$ represents the finite set of *actions*, s_0 is the *initial state*, and G describes the *goal* conditions. Each variable $v \in \mathcal{V}$ has a finite *domain* $\mathcal{D}_v$ if $v \in \mathcal{V}_p$, and $\mathcal{D}_v = \mathbb{Z}$ otherwise.[1] A *state* $s = \langle s_p, s_n \rangle$ is a complete assignment to the variables in $\mathcal{V}$, where $s_p \in \prod_{v \in \mathcal{V}_p} \mathcal{D}_v$ and $s_n \in \prod_{v \in \mathcal{V}_n} \mathbb{Z}$. By $s[v]$ we denote the value of the variable v at the state s. *Conditions* in an RT task can either be propositional or numeric. A propositional condition, also called *atom*, is of the form $\langle v, d \rangle$, where $v \in \mathcal{V}_p$ and $d \in \mathcal{D}_v$. Numeric conditions are (in)equalities of the form $v \bowtie w$ with $\bowtie \in \{\leq, =, \geq\}$, $v \in \mathcal{V}_n$ and $w \in \mathbb{Z}$.

State s satisfies a condition ψ if the corresponding constraint is fulfilled, denoted $s \models \psi$. We say that $s \models \langle v, d \rangle$ if $s[v] = d$, and $s \models v \bowtie w$ if $s[v] \bowtie w$. A set of conditions Ψ is satisfied by a state s, written $s \models \Psi$, if every condition in Ψ is satisfied. An *action* a is defined as a tuple $\langle \mathsf{pre}(a), \mathsf{eff}(a), \mathsf{cost}(a) \rangle$ where $\mathsf{pre}(a)$ and $\mathsf{eff}(a)$ are called preconditions and effects, respectively. Preconditions are a set of conditions that must be satisfied by a state in order for action a to be applicable, i.e., $s \models \mathsf{pre}(a)$. The result of action application is denoted $s' = s[\![a]\!]$. *Effects* can be either propositional or numeric. Propositional effects are a set of atoms $\langle v, d \rangle$ with $v \in \mathcal{V}_p$ and $d \in \mathcal{D}_v$ that are true after applying the action, i.e., $s'[v] := d$. Numeric effects are of the form $v \mathrel{+}= m$ with $v \in \mathcal{V}_n$ and $m \in \mathbb{Z} \setminus \{0\}$; the result of the application of such an effect is given by $s'[v] := s[v] + m$. For all variables that are not affected by the action we have $s'[v] := s[v]$. Each action modifies each variable at most once. Actions have non-negative costs $\mathsf{cost}(a) \in \mathbb{R}_0^+$. The goal conditions G define the set of goal states S_*, where $s_* \in S_*$ iff $s_* \models G$. An *s-plan* is a sequence of actions that are consecutively applicable in s and lead from the state s to a goal state, with the plan cost being the sum of action costs. An *optimal* s-plan minimizes this cost. An (optimal) plan for Π is an (optimal) s_0-plan. A heuristic $h : S \to \mathbb{R}_0^+ \cup \{\infty\}$ estimates the goal cost of state s, i.e., the cost of a cheapest plan for s. $h^*(s)$ denotes the cost of an optimal plan for s. If no goal state is reachable from s, then $h^*(s) = \infty$. A heuristic h is *admissible* if $h(s) \leq h^*(s)$ for all states $s \in S$.

Domain Abstractions in Classical Planning

We deal with transition systems (TS) that are induced by a planning task Π. A TS is a tuple $\mathcal{T} = \langle S, L, \mathsf{cost}, T, s_0, S_* \rangle$, where S is the set of states, L is the set of action labels of $\mathcal{A}$, cost is the action cost function, $T \subseteq S \times S \times L$ is the set of transitions, s_0 is the initial state, and S_* is the set of goal states. If the task is not obvious from context we write $\mathcal{T}_\Pi$. A map $\alpha : S \to S'$ induces the following TS $\mathcal{T}' = \langle \alpha(S), L, \mathsf{cost}, T', \alpha(s_0), \alpha(S_*) \rangle$, where $T' := \{(\alpha(s), \alpha(t), l) \mid (s, t, l) \in T\}$. We call α a state abstraction, and it can be seen as a special form of a graph homomorphism. The heuristic $h^\alpha(s) := h^*_{\mathcal{T}'}(\alpha(s))$ is admissible due to the homomorphic properties of α.

Domain Abstractions

In classical planning, an abstraction α is a domain abstraction if $\alpha := \{\alpha_v \mid v \in \mathcal{V}\}$, where each map $\alpha_v : \mathcal{D}_v \to 2^{\mathcal{D}_v}$ assigns to every $d \in \mathcal{D}_v$ a block $\alpha_v(d) \ni d$ such that the image $\{\alpha_v(d) \mid d \in \mathcal{D}_v\}$ forms a partition of $\mathcal{D}_v$. Abstract states are defined by $\alpha(s) := \{\langle v, \alpha_v(d) \rangle \mid \langle v, d \rangle \in s\}$. Note that multiple states can be mapped to the same abstract state. We denote abstract states by s_α. Domain abstractions are a special case of *Cartesian abstractions* (Seipp and Helmert 2018), which additionally allow the per-variable partition to depend on the values of other variables.

CEGAR

CEGAR is an iterative method for refining abstractions based on *flaws* (Seipp and Helmert 2018). Assume an abstract TS $\mathcal{T}'$ that results from an abstraction α applied to the concrete TS $\mathcal{T}_\Pi$ of a planning task Π. In each iteration, an optimal plan π for $\mathcal{T}'$ is computed. That plan is validated on the concrete state space, checking if some action a is inapplicable due to a *precondition flaw* $\psi \in \mathsf{pre}(a)$, i.e., a precondition ψ unsatisfied after applying the prefix of π up to a. In the absence of precondition flaws, goal flaws are computed. The set of *goal flaws* consists of all unsatisfied goal conditions after application of the abstract plan π in Π. If no flaws are found, then π is an optimal plan for Π and the problem is solved. Otherwise, a flaw $\langle v, d \rangle$ is refined as $\alpha_v(d) := \{d\}$ and $\alpha_v(d') := \alpha_v(d') \setminus \{d\}$ for all $d' \neq d$. By construction, CEGAR for domain abstractions is guaranteed to prevent re-occurring flaws once a flaw has been refined. Thereby, the abstract plans computed in each iteration are guaranteed to converge to an optimal plan for Π. Typically, a stopping condition for refinement is used, upper-bounding the number of abstract states by a constant.

Kreft et al. (2023) build on Rovner, Sievers, and Helmert (2019) to employ a *blacklisting* mechanism in CEGAR. The algorithm blacklists random subsets of variables after a specified percentage of the time limit has elapsed. Since there are far more possible abstractions than patterns, duplication of domain abstractions is rare.

Combining Abstractions Admissibly

A common approach to combine multiple (abstraction) heuristics to obtain an admissible aggregation is the canonical heuristic (Haslum et al. 2007). For a set of admissible heuristics H, for any state s, a maximum over these heuristics is an admissible estimate. A stronger combination can be obtained by summing over the heuristic values, but this is not admissible in general. We say that H is *additive* if $\sum_{h \in H} h(s)$ is admissible. A sufficient condition for the additivity of two abstraction heuristics $h^{\alpha_1}, h^{\alpha_2}$ is that no action $a \in \mathcal{A}$ induces a non-self-loop transition in both abstract TSs $\mathcal{T}^{\alpha_1}$ and $\mathcal{T}^{\alpha_2}$; in this case we call them *compatible*. The *conflict graph* on H has an edge between every pair

[1] The RT formalism usually allows real-valued numeric variables. We assume integer values in this paper, which is not a restriction, since every RT task can easily be compiled to integers.

of incompatible heuristics, and every independent set in this graph is therefore guaranteed to be additive. Summing the heuristic values within each maximum independent set and taking the maximum over all of them yields the canonical heuristic, which is admissible.

Abstracting Numeric Variables

We define a *numeric domain abstraction* as a pair of functions $\alpha := \langle \alpha^p, \alpha^n \rangle$ where α^p is the domain abstraction of finite-domain variables, and $\alpha^n := \langle \alpha^n_v \mid v \in V_n \rangle$ is a tuple of maps $\alpha^n_v : \mathbb{Z} \to 2^{\mathbb{Z}}$ that assign to every $d \in \mathbb{Z}$ a block $\alpha^n_v(d) \ni d$ of the form $[a, b] \cap \mathbb{Z}$, with $a, b \in \mathbb{Z} \cup \{-\infty, \infty\}$, such that the image $\{\alpha^n_v(d) \mid d \in \mathbb{Z}\}$ forms a partition of $\mathbb{Z}$. Blocks may be unbounded above or below, so a single map can cover the entire set of integers. Like in the classical setting, α remains a homomorphism: under the relaxed condition evaluation introduced below, every concrete transition $s \xrightarrow{a} t$ has a corresponding abstract transition $\alpha(s) \xrightarrow{a} \alpha(t)$ of equal cost, so $h^\alpha(s) := h^*_{\mathcal{T}'}(\alpha(s)) \leq h^*(s)$ is admissible independent of whether $\mathcal{D}_v$ is finite.

Ambiguous Numeric Condition Evaluations

Assume a numeric condition $v \bowtie w$ evaluated in an abstract state s_α. The result of such an evaluation is a priori undefined because $s_\alpha[v]$ is an integer interval, not a single value. We extend the evaluation of conditions such that $s_\alpha \models v \bowtie w$ iff there exists $d \in s_\alpha[v]$ s.t. $d \bowtie w$. This relaxes the evaluation of conditions under α. If $s_\alpha[v]$ contains both values that satisfy the condition and values that do not, the condition is considered *ambiguous* on s_α.

CEGAR for Numeric Conditions

To adopt the refinement strategy for finite-domain variables, we need a way to handle numeric conditions and effects. This is not straightforward. Consider a condition $v \bowtie w$ and a value $d \in \mathbb{Z}$ that does not satisfy it. Let α^n_v be the corresponding abstraction, and assume that $\alpha^n_v(d) \models v \bowtie w$. This means that there exists $x \in \alpha^n_v(d)$ such that $x \bowtie w$. Since $\alpha^n_v(d)$ is an integer interval, we can split it at d into two sub-intervals, one where $v \bowtie w$ does not hold for any value, and the other where $v \bowtie w$ holds at least one value.

A common splitting strategy in classical CEGAR partitions a domain into one block where the condition holds and another where it does not hold. In our numeric setting, an analogous split may leave one block where the condition holds and another where it either holds or is ambiguous, since intervals can mix satisfying and non-satisfying values.

For example, for the condition $v \geq w$ assume that $d < w$. Let $\alpha^n_v(d) := [a, b]$ be an integer interval.[2] Then, we can split $\alpha^n_v(d)$ into $[a, d]$ and $[d + 1, b]$. For every $y \in [a, d]$, the condition $y \geq w$ does not hold since $y \leq d < w$, so we eliminate the ambiguity. The $v \leq w$ case is symmetric. For the condition $v = w$, splitting $[a, b]$ at d does not fully resolve the ambiguity: the condition remains ambiguous on the sub-interval that contains w whenever it contains at least two values, while it does not hold on the other sub-interval.

[2] a and b can take the values $-\infty$ and ∞, respectively.

Non-deterministic Numeric Effects

Similarly to numeric conditions, numeric effects can also be ambiguous. Assume an abstraction α^n_v whose domain partition is $\mathcal{D}(\alpha^n_v) := \{[-\infty, 2], [3, 3], [4, \infty]\}$ and an action a with numeric effect $v \mathrel{+}= 1$ and no preconditions. If the current abstract state has $s_\alpha[v] = [-\infty, 2]$, then $d = 1$ and $d = 2$ are both valid values for v in the concrete TS, and the application of a may result in transitions to the two intervals $[-\infty, 2]$ and $[3, 3]$. We support these non-deterministic abstract transitions, which cannot occur in classical planning, by creating a copy of the action for each transition, extending preconditions and effects with partition indices.

Note that any action with numeric effects of the form $v \mathrel{+}= c$ can be viewed as having infinitely many conditional effects of the form $\langle v, d \rangle \triangleright \langle v, d + c \rangle$ for each $d \in \mathbb{Z}$. This motivates treating the non-deterministic transitions induced by constant additive effects as *deviation flaws*, previously introduced by Seipp and Helmert (2018) for classical planning with conditional effects and later adopted by Büchner et al. (2024) for domain abstractions on factored SAS$^+$ tasks.

In the context of CEGAR, non-determinism means that an abstract plan may deviate from the corresponding concrete execution. Let $\tau^\alpha = s^0_\alpha \xrightarrow{a_1} \ldots \xrightarrow{a_n} s^n_\alpha$ be an abstract plan trace with $s^0_\alpha := \alpha(s_0)$ and $s^n_\alpha \in \alpha(S_*)$. Executing $[a_1, \ldots, a_n]$ from the concrete initial state s_0 yields a partial trace $s_0 \xrightarrow{a_1} \ldots \xrightarrow{a_i} s_i$, where, due to non-determinism in the abstraction, we may reach a state s_i such that $\alpha(s_i) \neq s^i_\alpha$, which is in fact a deviation flaw.

Consider another example with a variable v with $s_0[v] = 3$ and $\mathcal{D}(\alpha^n_v) = \{[-\infty, 5], [6, \infty]\}$. Let an action $inc(v)$ have the effect $v \mathrel{+}= 1$. An abstract plan $[inc(v)]$ yields the trace $\{\langle v, [-\infty, 5] \rangle\} \xrightarrow{inc(v)} \{\langle v, [6, \infty] \rangle\}$. However, the concrete execution yields $s_0 := \{\langle v, 3 \rangle\} \xrightarrow{inc(v)} s_1 := \{\langle v, 4 \rangle\}$, so $\alpha(s_0[v]) = \alpha(s_1[v]) = [-\infty, 5]$, which deviates from the expected state $\alpha(s_1[v]) = [6, \infty]$. To handle such deviation flaws, we split v at the value it has before applying $inc(v)$. More formally, a *deviation flaw* occurs for a transition $s_\alpha \xrightarrow{a} s'_\alpha$ in an abstract trace τ^α if there exists a concrete transition $s \xrightarrow{a} s'$ with $\alpha(s) = s_\alpha$ but $\alpha(s') \neq s'_\alpha$.

Overall, we deal with deviation flaws as just described, treat *precondition flaws*, where an action a_i cannot be executed in s_{i-1} because $s_{i-1} \not\models \mathsf{pre}(a_i)$, and *goal flaws*, where the final state of the execution of τ^α is not a goal state, i.e., $s_n \not\models G$, which is in line with classical planning.

Theorem 1. *The proposed CEGAR procedure is semi-complete on* RT *tasks without zero-cost actions.*

Proof Sketch: Let L be the cost of an optimal concrete plan (assumed to exist). Since actions have positive cost, the states and numeric values reachable within cost L form a finite sub-transition system $\mathcal{T}'$. Refinement splits intervals only at concrete values from this finite fragment, so only finitely many abstractions are possible, and each split either eliminates a flaw witness or forces the abstract plan cost to grow (bounded by L). Hence refinement terminates with an abstraction whose cheapest plan has cost L and no remaining flaws within that cost, yielding a valid concrete plan. □

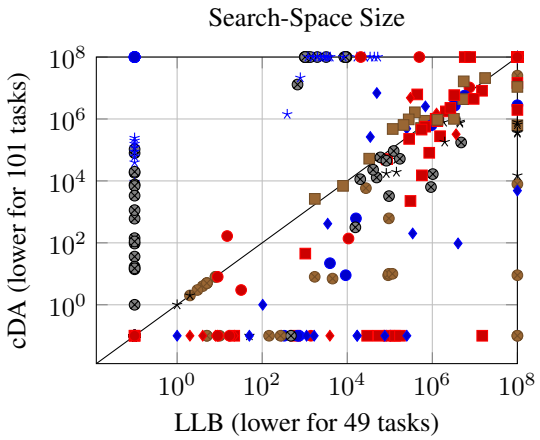

Figure 1: Expansions until the last f-layer, comparing our cDA (y-axis) to PDBs in the LLB variant (x-axis).

Experimental Evaluation

We implemented our domain abstractions and CEGAR in Numeric Fast Downward (Aldinger and Nebel 2017). For our empirical evaluation we used a cluster of Intel Xeon Gold 6130 CPUs, Downward Lab 8.2 (Seipp et al. 2017), and the 30 min / 8 GiB limits of IPC 2023 (Taitler et al. 2024) on the benchmark set of Gnad et al. (2025). We compare a single domain abstraction (**sDA**) and the canonical heuristic over multiple abstractions (**cDA**) to numeric Pattern Database (PDB) heuristics in variants **BBB** (Gnad et al. 2025) and **LLB** (Fritzsche et al. 2026), and LM-cut (**LMc**) (Kuroiwa et al. 2022); all heuristics use A^* search.

Parameter Study

Following Kreft et al. (2023), we sweep parameters preliminarily, varying the collection-generation time (100–700 s), abstraction size (10^4–10^6 states), collection size (10^5–10^7), and blacklist-trigger percentage (0%–60%) for excluding numeric and propositional variables from refinement. We additionally vary the initialization and refinement settings: starting with a random or goal (G) variable; fully refining it (I; for numeric variables we split at the initial-state value) or refining an arbitrary atom; selecting flaws randomly or via minimum abstract state-space growth (MINGROWTH); and collecting all flaws along an abstract plan execution rather than stopping at the first (Pozo, Torralba, and Linares López 2024). The best-performing configuration uses 1M abstraction size, 10M collection size, 700 s of generation time, 60% blacklist trigger, and the MINGROWTH+GI combination – consistent with Kreft et al. (2023) on the qualitative pick, although unlike them we observed a high impact of the blacklist percentage and abstraction size. The lowest-coverage configuration we tested solved 221 instances. BBB and LLB use the smaller per-abstraction defaults of Fritzsche et al. (2026) (10^5/10^6), with the same 8 GiB limit; cDA therefore operates with a roughly 10× larger abstraction-size budget. Note that we tuned cDA on the evaluation benchmarks while the PDB baselines were not similarly tuned; results should be interpreted with this asymmetry in mind.

	Domain	#	Blind	LMc	BBB	LLB	sDA	cDA
Restricted Task (RT)	delivery	20	2	**3**	2	2	2	**3**
	depots	20	4	**7**	**7**	**7**	**7**	**7**
	depots-sym	20	4	**7**	6	6	6	6
	drone	20	3	3	**4**	**4**	3	**4**
	expedition	20	5	**6**	**6**	**6**	**6**	**6**
	forestfire	20	10	**11**	10	10	10	**11**
	MC-pogo	20	14	5	18	17	**19**	**19**
	MC-sword	20	**20**	9	**20**	**20**	19	**20**
	mprime	20	6	**15**	12	12	13	14
	rover-unit	20	4	**7**	6	6	4	6
	sugar	20	2	**12**	3	3	6	8
	zenotravel	23	6	**13**	10	10	8	8
	zenotravel-ipc	20	6	**8**	6	6	6	7
	others	40	0	0	0	0	0	0
	RT Sum	303	86	106	110	109	109	**119**
Compiled to RT	counters	20	3	**5**	**5**	**5**	3	4
	counters-sym	11	2	**11**	9	9	2	2
	farmland	30	12	**30**	12	26	12	12
	farmland-ipc	15	4	**15**	8	**15**	4	4
	fn-counters-sm	8	6	**7**	**7**	**7**	6	**7**
	hydropower	20	9	**10**	9	9	**10**	**10**
	petri-net	20	2	8	9	8	4	**10**
	sailing	40	10	**40**	15	15	11	16
	sailing-ipc	20	0	**8**	0	0	0	0
	satellite	20	1	2	2	1	**3**	**3**
	others	103	68	68	68	68	68	68
	Compiled Sum	307	117	**204**	144	163	123	136
	Total Sum	610	203	**310**	254	272	232	255

Table 1: Coverage results, comparing the blind and LMc baselines, and PDBs to our domain-abstraction variants.

Table 1 reports coverage. Overall, our methods (sDA and cDA) are competitive with plain numeric PDBs (BBB). On the RT-fragment domains (top half), cDA solves the most instances, even beating the numeric LM-cut heuristic, although performance varies across individual domains. On non-RT domains compiled to RT (bottom half), cDA falls behind; disentangling the role of the compilation from other factors is left to future work. In terms of heuristic accuracy, our domain abstractions are complementary to PDBs and often provide better search guidance (Figure 1).

Conclusion

We generalize domain abstractions from classical to simple numeric planning, guiding the construction of the abstraction with an adapted CEGAR algorithm. To obtain an informative heuristic, we combine multiple abstractions with the canonical heuristic. Our approach is competitive with state-of-the-art heuristics like LM-cut on natively RT-modeled domains, while falling behind on domains compiled to RT. Smarter flaw selection and numeric cost partitioning are promising directions for future work. Another interesting question is how domain abstractions and CEGAR can be adapted for a richer fragment of numeric planning, e.g. allowing for linear expressions in preconditions and effects.

Acknowledgements

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725. Alexander Shleyfman's work was partially supported by ISF grant 2443/23.

References

Aldinger, J.; and Nebel, B. 2017. Interval Based Relaxation Heuristics for Numeric Planning with Action Costs. In *SOCS*, 155–156.

Büchner, C.; Ferber, P.; Seipp, J.; and Helmert, M. 2024. Abstraction Heuristics for Factored Tasks. In *ICAPS*, 40–49.

Clarke, E. M.; Grumberg, O.; Jha, S.; Lu, Y.; and Veith, H. 2000. Counterexample-Guided Abstraction Refinement. In *CAV*, 154–169.

Culberson, J. C.; and Schaeffer, J. 1998. Pattern Databases. *Comp. Intell.*, 14(3): 318–334.

Edelkamp, S. 2001. Planning with Pattern Databases. In *ECP*, 13–24.

Fritzsche, M.; Gnad, D.; Gruntov, M.; and Shleyfman, A. 2026. Managing Infinite Abstractions in Numeric Pattern Database Heuristics. In *AAAI*, 36227–36235.

Gnad, D.; Alon, L.-O.; Weiss, E.; and Shleyfman, A. 2025. PDBs Go Numeric: Pattern-Database Heuristics for Simple Numeric Planning. In *AAAI*, 26507–26515.

Gnad, D.; Helmert, M.; Jonsson, P.; and Shleyfman, A. 2023. Planning over Integers: Compilations and Undecidability. In *ICAPS*, 148–152.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-Independent Construction of Pattern Database Heuristics for Cost-Optimal Planning. In *AAAI*, 1007–1012.

Helmert, M. 2002. Decidability and Undecidability Results for Planning with Numerical State Variables. In *AIPS*, 303–312.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge-and-Shrink Abstraction: A Method for Generating Lower Bounds in Factored State Spaces. *J. ACM*, 61(3): 16:1–16:63.

Hoffmann, J. 2003. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. *JAIR*, 20: 291–341.

Kreft, R.; Büchner, C.; Sievers, S.; and Helmert, M. 2023. Computing Domain Abstractions for Optimal Classical Planning with Counterexample-Guided Abstraction Refinement. In *ICAPS*, 221–226.

Kuroiwa, R.; Shleyfman, A.; Piacentini, C.; Castro, M. P.; and Beck, J. C. 2022. The LM-Cut Heuristic Family for Optimal Numeric Planning with Simple Conditions. *JAIR*, 75: 1477–1548.

Li, D.; Scala, E.; Haslum, P.; and Bogomolov, S. 2018. Effect-Abstraction Based Relaxation for Linear Numeric Planning. In *IJCAI*, 4787–4793.

Pearl, J. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley.

Pozo, M.; Torralba, Á.; and Linares López, C. 2024. Gotta Catch 'Em All! Sequence Flaws in CEGAR for Classical Planning. In *ECAI*, 4287–4294. IOS Press.

Rovner, A.; Sievers, S.; and Helmert, M. 2019. Counterexample-Guided Abstraction Refinement for Pattern Selection in Optimal Classical Planning. In *ICAPS*, 362–367.

Scala, E.; Haslum, P.; Thiébaux, S.; and Ramírez, M. 2020. Subgoaling Techniques for Satisficing and Optimal Numeric Planning. *JAIR*, 68: 691–752.

Seipp, J. 2017. Better Orders for Saturated Cost Partitioning in Optimal Classical Planning. In *SOCS*, 149–153.

Seipp, J.; and Helmert, M. 2018. Counterexample-Guided Cartesian Abstraction Refinement for Classical Planning. *JAIR*, 62: 535–577.

Seipp, J.; Keller, T.; and Helmert, M. 2020. Saturated Cost Partitioning for Optimal Classical Planning. *JAIR*, 67: 129–167.

Seipp, J.; Pommerening, F.; Sievers, S.; and Helmert, M. 2017. Downward Lab. https://doi.org/10.5281/zenodo.790461.

Taitler, A.; Alford, R.; Espasa, J.; Behnke, G.; Fiser, D.; Gimelfarb, M.; Pommerening, F.; Sanner, S.; Scala, E.; Schreiber, D.; Segovia-Aguas, J.; and Seipp, J. 2024. The 2023 International Planning Competition. *AI Mag.*, 45(2): 280–296.